International Journal of Applied Metaheuristic Computing
Volume 13 • Issue 1

Software Vulnerability Prediction Using Grey Wolf-Optimized Random Forest on the Unbalanced Data Sets

Wasiur Rhmann, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, India*

ABSTRACT

Any vulnerability in the software creates a software security threat and helps hackers to gain unauthorized access to resources. Vulnerability prediction models help software engineers to effectively allocate their resources to find any vulnerable class in the software before its delivery to customers. Vulnerable classes must be carefully reviewed by security experts and tested to identify potential threats that may arise in the future. In the present work, a novel technique based on grey wolf algorithm and random forest is proposed for software vulnerability prediction. Grey wolf technique is a metaheuristic technique, and it is used to select the best subset of features. The proposed technique is compared with other machine learning techniques. Experiments were performed on three datasets available publicly. It was observed that the proposed technique (GW-RF) outperformed all other techniques for software vulnerability prediction.

KEYWORDS

Grey Wolf, Machine Learning Techniques, Prediction, Random Forest, Vulnerability

1. INTRODUCTION

A vulnerability is a weakness in the software that, when exploited, causes a security failure. Due to time constraints, software developers usually do not concern much about the security aspects at the initial stages of the software development that results in security failures in the operational stages. It is difficult to detect the vulnerability in the software until they hinder the normal operation of the software. Prediction of software vulnerability during the early stage of the life cycle is a promising approach. Software organizations perform security checks to avoid software failures and the presence of vulnerabilities in the software may lead to software failures. A fault in the software specification, development, or its configuration is vulnerability if its execution results in a violation of security policy (McGraw & Potter, 2004). A fault in the software system if accidentally executed then the software may not be able to perform its required or expected function (Shin & Williams, 2008). Software faults are defects or bugs in the software system and vulnerability refers to those software faults which leads security failure if exploited. Software metrics are heavily used in literature to predict software maintainability and change (Bansal, 2017) and defect proneness (Gyimothy et al., 2005) Numerous studies have shown the relation between software architecture and structural software metrics like complexity, coupling, and cohesion (CCC).CCC metrics are very efficient in measuring the quality of software architecture (QSA) and QSA influences the quality of software. Despite being heavily used of these metrics there is no available proper guideline on how one can

DOI: 10.4018/IJAMC.292508 *Corresponding Author

use these structural metrics in the prediction of the software vulnerability. The use of these structural metrics in vulnerability prediction may lead to more secure and reliable software (Walden, et al.,2014) . Early-stage detection of security vulnerabilities in the software development life cycle may mitigate the risk of software security failures. In recent years a number of metaheuristics algorithms like Particle swarm optimization(PSO), Genetic algorithm(GA), Firefly algorithm(FA) etc are applied for feature selection and hyper-parameter optimization of machine learning and deep learning algorithms. Neggaz et al. (2020a) have used a novel Henry gases solubility optimization for feature selection. Proposed Henry gases solubility optimization is compared with six other algorithms on 12 datasets. This technique has shown improved accuracy with less number of features. Neggaz et al. (2020b) have proposed improved slap swarm algorithm for feature selection. They have used Sine Cosine algorithm and Disrupt Operator and compared accuracy with other swarm intelligence algorithms and found better results. In our previous work(Rhmann et al., 2021) metaheuristic algorithms firefly algorithm, genetic algorithm and black hole optimization algorithms are used for optimization of software efforts using ensemble techniques. In this study metaheuristic algorithms based random forest techniques are applied for software vulnerability prediction.

The rest of the paper is organized as follows: Section 2 describes the related work Section 3 describes the data-sets used in the study. Section 4 describes the proposed software vulnerability model. Section 5 describes the performance evaluation measures. In Section 6 experimental setup and results of prediction models are given. In section 7 comparative analysis of the work is presented and section 8 describes the application of work. Section 9 describes the threats to the validity and finally, section 10 is used to conclude the work.

2. RELATED WORK

Various MLT and statistical techniques are used to predict software quality. Different structural metrics are significantly contributing to the prediction of software quality attributes. These structural metrics can be also helpful in the prediction of software vulnerability. Chowdhury et al. (2011) have proposed a framework that used the cyclomatic complexity metric to predict the vulnerabilities. Vulnerability prediction models are constructed using different data mining techniques like C4.5 Decision Tree, Random Forests, Logistic Regression, and Naïve Bayes. Alenezi and Abunadi(2016) compared the performance of classification techniques in the detection of vulnerable files. Three large open-source web projects are used in an empirical study for the investigation of vulnerable code using software metrics. With the detection of vulnerable code software professionals can prioritize verification efforts. It was observed that software metrics are helpful in the prediction of vulnerabilities and detection of vulnerable code is helpful in security auditing efforts. Walden et al. (2014) compared vulnerability prediction models based on text mining and models based on software metrics. It was observed that the prediction rate of models based on text mining was better compared to the models based on software metrics. Basili et al. (1999) used the OO metrics for the prediction of fault-prone classes. Eight medium size projects are used for the collection of data for fault prediction and technique was used for fault prediction. It was observed that 88% of faulty classes were able to predict with 60% precision. Menzies et al. (2007) performed experiments on NASA repository to predict defects using software metrics. Prediction models were built using the J48, Naive Bayesian model. The authors detected 71% of defects with a 25% false-positive rate. Alenezi et al. (2014) predicted faulty classes with Naive Bayes, Bayesian Networks, J48, and Random Forests techniques. 92% of the defects were detected using Random Forest models. Various soft computing and search based algorithm (SBA) are also found to be used in the development of models for prediction of faults (Gyimothyetetal, 2005; Catal, 2007; Vandecruys et al., 2008; Carvalho et al, 2010; Pendharka, 2010) and efforts ((Shukla, 2000; Burgess and M. Lefley 2001; Bardsiri,2013; Minku and Yao; 2013) . Bozorgi et al. (2010) have predicted the exploitation of vulnerability using publicly available databases. They trained their model with data features like text field, time, stamp, and other entries available in the vulnerability report.

Their model performed better in comparison to the available standard to predict vulnerability based on expert knowledge and heuristics. Shar et al. (2015) have built supervised and semi-supervised learning based vulnerability prediction models. They performed experiments on seven open-source projects with vulnerabilities like SQL injection, cross-site scripting, remote code execution, etc. and it was observed that semi-supervised model have better recall and low probability of false alarm with the low amount of labeled vulnerability data.

3. DATASETS USED IN THE STUDY

In this study, we have used PHP data-sets due to wide acceptance of PHP as the server-side programming language in the web development and it is open source and known for its poor security reputation (Chowdhury, et al.,2011) PHP data-set is collected from open source community by (Waldenetal,2014). Open-source software is usually considered secure due to the involvement of large numbers of the developer to detect and fix vulnerabilities in software code (Meneely and Williams, 2010). The data set contains the vulnerability information of PHP files along with the software metrics. The data-set contains different applications which are Drupal6, Moodle2, and phpMyAdmin3.3. Drupal6 is a content management system. Moodle is an open-source learning management system. phpMyAdmin3.3 is a free web-based management tool to write web applications in PHP for MySQL database. Descriptive statistics about the data-set are given in Table 1. Followings are the software metrics which are included in this data-set are:

Lines of code: Number of lines in a PHP source. It includes PHP tokens and excludes the lines without PHP tokens.

Lines of code (non-HTML LOC): It is defined as the number of lines of code skipping the HTML content embedded in PHP files.

Number of functions: It is calculated as the number of functions and method definitions in a PHP file.

Cyclomatic complexity: The control flow graph of the program gives the cyclomatic complexity of the program. It can be also computed by adding one to the decision statements in the PHP file.

Maximum nesting complexity: The Maximum nesting complexity refers to the maximum depth up to which loops and control structures in the php data file are nested.

Halstead's volume: In the calculation of this metric the number of unique operators and operands and the number of total operators and operands in the file are used. This metric is derived with the consideration of method names and PHP language operators as operators, and parameter and variable names are operands.

Total external calls: It is defined as the number of instances where a statement in the file which is being measured invokes a function defined in a different file.

Fan-in: It is measured as the number of files that contain statements that invoke a function or method defined in the file of which fan-in is being measured.

Fan-out: It is measured as the number of files that contain functions or methods invoked by statements in the file of which fan-out is being measured.

Internal functions or methods called: It is calculated as the number of functions or methods defined in the file of which it is being measured, which are called at least once by a statement in the same file.

External functions or methods called: It is calculated as the number of functions or methods defined in other files which are called at least once by a statement in the file of which it is being measured.

External calls to functions or method: It is calculated as the number of files calling a particular function or method defined in the file of which it is being measured summed across all functions and methods in the same file.

Table 1. descriptive statistics about the data-set

Data-set	Vulnerable Files	Total Files
Drupal 6	62	202
Moodle 2	24	2942
phpMyAdmin3.3	27	322

4. PROPOSED PREDICTION MODELS

In this section smote technique, grey wolf algorithm, random forest and proposed technique for the creation of vulnerability prediction model are described.

4.1 SMOTE

Oversampling is used to tackle the problem of an imbalanced dataset. The minority samples are synthetically generated to balance the datasets using augmentation to original datasets. This technique is called Synthesized Minority Oversampling Technique(Chawla, 2002). K-nearest neighbors are generated for each instance of the minority class. Based on the imbalanced proportion of datasets N samples are selected from the k-nearest neighbor of each instance and collection of these is a new Set A. Then for each member of set A new instances are generated using:

$$x'=x+rand(0,1)*|x-x'|$$

where rand(0,1) gives a random number between 0 and 1.

4.2 Metaheuristic Algorithms

Metaheuristic Algorithms are search based optimization algorithms and are very helpful to find optimal solutions when getting exact optimal solution with available resources is difficult to achieve. These algorithms are like a template which can be used to find optimal solutions of different problems.

4.3 Grey Wolf Algorithm

Grey wolf algorithm is inspired by the nature of grey wolves(Mirjalil, 2014). Grey wolves leadership and haunting mechanism help to design a new metaheuristic algorithm with three steps: searching prey, encircling prey, and attacking prey.

The wolves live in packs and there are around 12-15 in a group. These wolves have a hierarchical management system. α wolves are the team leader and strongest and direct all other wolves in the group. β and γ are next to α and their task is to support α. Ω is at the bottom of the hierarchy and these wolves are in large quantity they are responsible for the internal relationship.

$$D_i=|C.\ Xp(t)-X(t)|$$
$$X(t+1)= Xp(t)-A.Dp$$

X(t) and X(t+1) are the current positions of the grey wolf and its next position. t is the iteration number. A and C are coefficient vectors computed as:

$$A=2ar_1-a$$
$$C=2r_2$$

r_1, r_2 are random numbers with values between 0 and 1.

$$a=2-2t/t_{max}$$

where t_{max} is the maximum iteration number. Values of a decrease from 2 to 0.

The movements of omega wolves are guided by the position of α, β and γ wolves. The movements of these wolves are calculated using the following equations:

$D\alpha=|C. Xp(t)-X(t)|$
$D\beta=|C. Xp(t)-X(t)|$
$D\gamma =|C. Xp(t)-X(t)|$
$X_1= X\alpha (t)-A.D\alpha$
$X_2= X\beta (t)-A.D_\beta$
$X_3=X\gamma(t)-A.D\gamma$
$Xp(t+1)=(X_1 + X_2+ X_3)/3$

α, β and γ gives optimal solution, sub optimal solution and third optimal solution. Movements of all other grey wolves are guided by these three wolves as they are closed to the prey.

After surrounding the prey, grey wolves will capture it. $|A|<1$ then the grey wolf will capture the prey by approaching. $|A|\geq 1$ then grey wolves will move away from the prey and will do the global search.

4.4 Particle Swarm Optimization

PSO is a very simple stochastic optimization algorithm which doesn't require function to be differentiable(Kennedy and Eberhart, 1995). It is inspired from social behavior of birds. There are very few hyper parameters which make is very efficient for many applications. Initial population is generated randomly and after each iteration new population with better fitness is generated. Particles are moved in search space with a velocity and update its position according to their best position and swarm's best known position.

4.4.1 Genetic Algorithm(GA)

Genetic algorithm is biological evolution based optimization algorithm(Mitchell, 1998). This algorithm finds optimal solution by searching in solution space.

Following are the main steps to get optimal solution of a problem:

Initial population generation:

Initial population of candidate solutions are generated randomly and a function which measures the appropriateness of generated solutions is used. This function is called fitness function and it depends upon the problem objective

Selection of good solutions

Based on fitness of solutions individuals from populations are selected for crossover operation to generated new solutions

Crossover and mutation

For generation of new solutions crossover and mutation are used in crossover two individuals are combined while in mutation only some bits of individual solutions are changed

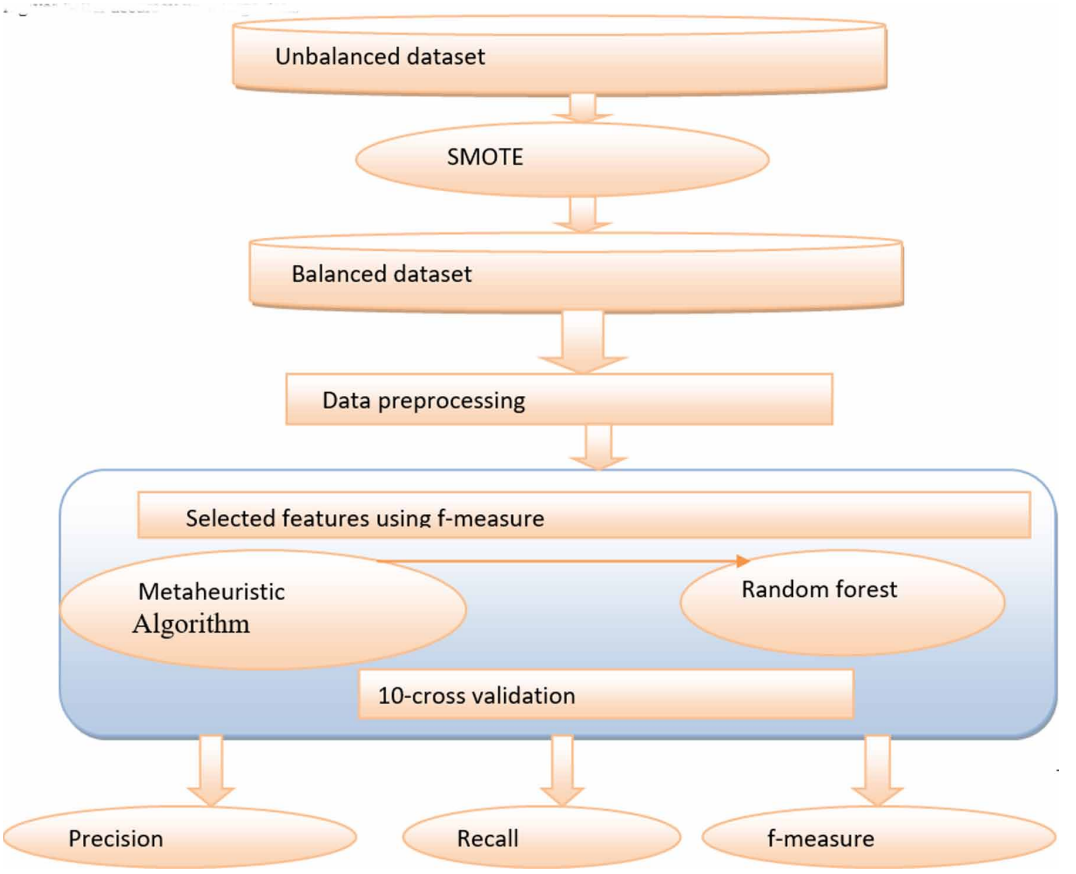
These steps are repeated till we get optimal solution or we have reached at a predefined number of generations.

4.5 Random Forest

Random Forest uses multiple algorithms to give better performance in which several decision trees (DT) are generated in training time and a class level is given by each tree. It is more robust to noise comparison to DT in the presence of inter correlated features. All complexity metrics are usually inter-correlated so we have selected the RF. It is expected to perform better than the decision tree. It can take input variables with binary, categorical, continuous features, and doesn't require feature scaling. It can handle over fitting efficiently. Although it is computationally complex it gives better accuracy with large data.

The proposed framework of software vulnerability prediction is shown in fig. 1. The three datasets used in the study are unbalanced as the numbers of vulnerable to non-vulnerable classes are unequal. To make the datasets balance SMOTE(Synthetic Minority Oversampling technique) is used. Then datasets are preprocessed and they are standardized to make all the features with mean of 0 and standard deviation 1

Figure 1. Proposed framework of software vulnerability prediction



Then 80% of the dataset in spitted in stratified manner is used for training the Random forest classifier and 10-cross validation is used to measure the performance of the classifier. Obtained f-measure is used as a fitness function for the metaheuritic algorithms. Then the metaheuristic algorithm finds the best features with the optimized precision, recall, and f-measures.

The main steps of the metaheuristic algorithm based Random forest algorithm are given as:

Table 2. Algorithm

Algorithm
Input: Dataset (D) **Output: Optimized cross-validated Results** **{Precision, Recall, f-measure}** **Step1: Initialize the initial population for Metaheuritic algorithms as:** **{a_1,a_2,a_3,…..,a_n}** **where a_i=[0,1]** **0 means feature is not selected** **1 means feature is selected** **and n is the number of features in the dataset.** **Step2: Take the Fitness function as** **F=f-measure of Random forest on a subset of D** **Step3: Repeat the process up to 2 steps until the desired number of iterations or we get maximum f-measure =1.**

The pseudo-code of the proposed Metaheuristics-RF is given below:

Table 3. Pseudo-code of metaheuristics-RF

Pseudo of Metaheuristics-RF:
Input: Vulnerability dataset=D **Output: Optimal values of f-measure, precision and recall** **Initialize the values: Number of dimension=independent features of dataset** **Number of generation=n, population size=N** **Take initial candidate solution as {a_1, a_2,a_3,…..,a_n} where a_i=[0,1] and n is the number of features in the dataset.** **For each iteration:** **Selection features set D' from D** **Divide D' into 80:20 ratio {D_{tr}, D_{te}}** **Preprocess and standardize the dataset D'** **Train a random forest (RF) with D_{tr}** **Evaluate the RF with 10-cross validation.** **Return fitness=** $\frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$ **After n generation or f-measure=1** **Best f-measure, precision and recall on 10-cross validation.** **End**

5. PERFORMANCE EVALUATION MEASURES

In our work, we have used the data of Drupal6, Moodle2, and phpMyAdmin3.3. We performed experiments with the aim to identify the technique which is more accurately predicts the software vulnerability. In the present study, positive cases are vulnerable classes while negative cases are non-vulnerable classes. As the number of vulnerable and non-vulnerable classes are unequal for each data-set Precision, recall and f-measure are used for measurement of performances. These measures are described as follows:

TP(True Positive) It is the number of vulnerable classes declared as vulnerable.
FP(False Positive) It is the number of non-vulnerable classes declared as vulnerable.
FN(False Negative) It is the number of vulnerable classes declared as non-vulnerable.

Precision

Precision is defined as the ratio of correctly predicted vulnerable classes to the total number of classes predicted as vulnerable by model.

$$\text{Precision}= \frac{TP}{TP + FP}$$

5.1 Recall

Recall is defined as the ratio correctly predicted vulnerable classes to total vulnerable classes in data.

$$\text{Recall}= \frac{TP}{TP + FN}$$

5.2 F-measure(F1-score)

For binary classification f-measure is simply harmonic mean of precision and recall.

Values of precision and recall, f-measure lie in between 0 and 1. Best values of both are 1.

6. EXPERIMENTAL SETUP AND RESULT

All the experiments are performed in python using python packages Scikit-learn (Pedregosa et al., 2011) and Niapy(Vrbancic et al., 2018). Machine learning algorithms used in the study are Support Vector Machine (SVM), Naïve bayes(NB), Gradient Boosting(GB), and Random forest. All Machine learning techniques are applied in the default setting. Metaheuristics algorithms used in the study are Grey wolf optimization, particle swarm optimization and genetic algorithm and these algorithms are implemented in Niapy package. For efficient working of prediction models, transformation is performed. Datasets are divided into 80:20 for training and testing purpose and 10-cross validations are used for measuring the performances of different techniques. For each technique precision, recall and f-measures are calculated on different datasets and presented in table 2.

The highest f-measure on phpmyadmin3.3 dataset is 0.959 by the PSO-RF technique. For moodle2 dataset highest f-measure is 0.990 obtained by PSO-RF and GW-RF. For drupal6 dataset highest f-measure is 0.869 for PSO-RF. Hence PSO-RF technique has shown the best performance on all three datasets.

Fig. 2, Fig. 3 and Fig. 4 have presented the graph of precision, recall, and f-measure of different techniques.

It is clear that precision on all datasets is achieved by the PSO-RF techniques and random forest is next to the PSO-RF on all datasets. Recall values of PSO-RF, random forest, and gradient boosting are very close on all datasets. From fig. 4 f-measure is best for the PSO-RF technique. MLT have shown f-measure next to the PSO-RF.

Table 4. Performances of different techniques

Data-set	Technique	Precision	Recall	F-measure
phpmya dmin3.3	SVM	0.74	0.59	0.66
	Naive Bayes	0.76	0.34	0.46
	GB	0.87	0.95	0.90
	Random Forest	0.90	0.97	0.94
	GW-RF	0.954	0.962	0.955
	PSO-RF	0.962	0.968	0.959
	GA-RF	0.951	0.95	0.955
moodle2	SVM	0.79	0.68	0.73
	Naive Bayes	0.80	0.35	0.48
	Gradient boosting	0.91	0.99	0.95
	Random Forest	0.96	1	0.98
	GW-RF	0.984	0.998	0.990
	PSO-RF	0.984	0.998	0.990
	GA-RF	0.982	0.998	0.988
Drupal 6	SVM	0.74	0.64	0.67
	Naive Bayes	0.81	0.41	0.52
	GB	0.76	0.88	0.81
	Random Forest	0.80	0.90	0.85
	GW-RF	0.841	0.921	0.865
	PSO-RF	0.83	0.93	0.869
	GA-RF	0.83	0.92	0.866

Figure 2. Precision of different techniques

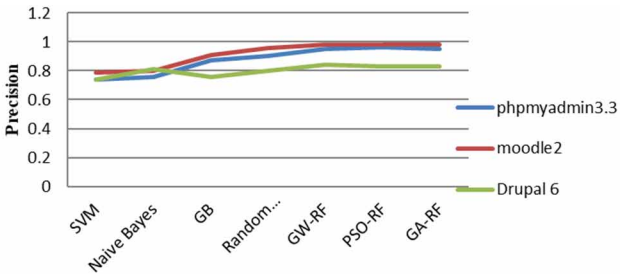


Fig. 5 is box-plot of the values of f-measure of different techniques and it is clear that GW-RF, PSO-RF and GA-RF have shown better f-measures compared to machine learning techniques.

7. COMPARATIVE ANALYSES WITH RELATED WORK

In the present section different studies that have used the datasets phpmyadmin3.3, moodl2 and drupal6 are compared with our proposed technique. Khalid et al. (2018) have predicted software vulnerability

Figure 3. Recall of different techniques

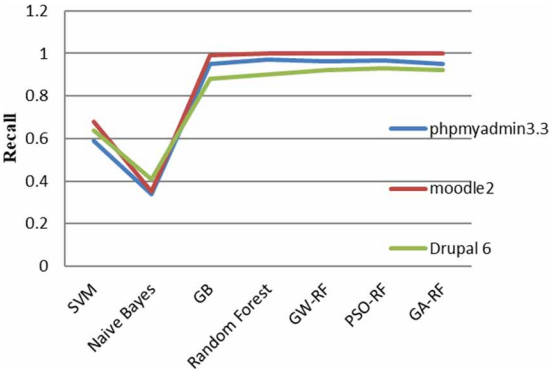


Figure 4. F-measure of different techniques

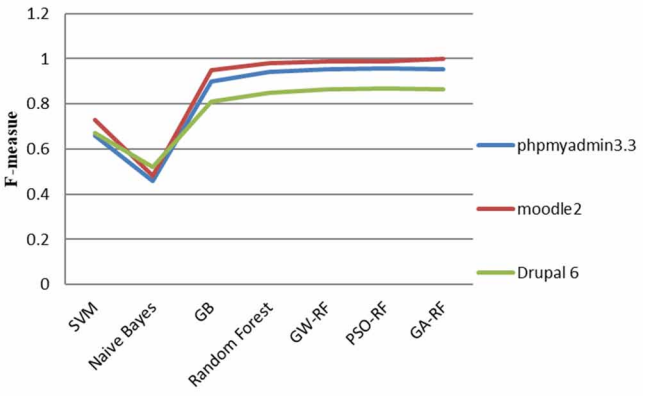
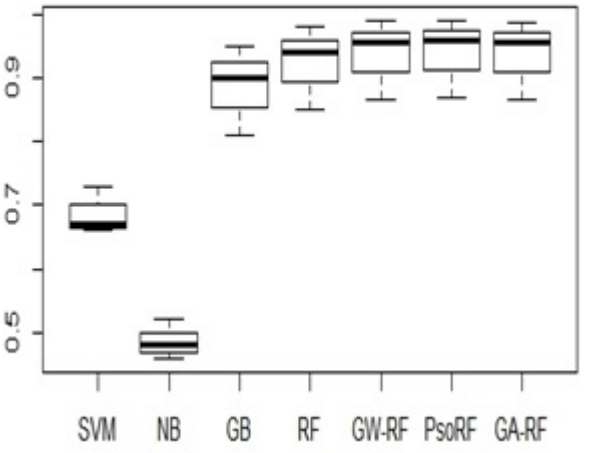


Figure 5. Box-plot of f-measure of different techniques for different datasets



on the same datasets phpmyadmin3.3, moodle2, and drupal7. They experimented with weka software and used random forest and decision trees and a meta classifier for the development of a prediction model. Abunadi et al. (2016) have performed experiment on the same web-based datasets as used in the study. They used weka software and applied machine learning techniques without considering balancing the datasets in 10 cross-validation manner. Zhang et al. (2015) have used the web datasets phpmyadmin3.3m moodle2 and drupal6 and applied ensemble techniques. Walden et al. (2014) have used the three same web datasets and used random forest as their main classifier with 100 forest in weka and compared their performance with text mining-based vulnerability prediction. The f-measures of these studies are compared with our proposed technique and results are given in Table 3.

Table 5. Comparative analysis of different techniques

Author and year	Data-set	Best technique	F-measure
Proposed	phpmy admin3.3	PSO-RF	0.959
	moodle2	GW-RF and PSO-RF	0.990
	Drupal 6	PSO-RF	0.869
Khalid et al. (2018)	phpmya dmin3.3	Meta classifier(Random forest)	0.463
	moodle2	Meta classifier (Decision tree(J48))	0.202
	Drupal 6	Meta classifier (Random forest)	0.848
Abunadi et al. (2016)	phpmya dmin3.3	Random forest	0.913
	moodle2	Random forest	0.991
	Drupal 6	Random forest	0.752
Zhang et al., (2015)	phpmy admin3.3 and text feature	Ensemble	0.340
	moodle2 and text feature	Ensemble	0.071
	Drupal 6 and text feature	Ensemble	0.683
Walden et al. (2014)	phpmy admin3.3	Random forest	0.227
	moodle2	Random forest	0.035
	Drupal 6	Random forest	0.562

7.1 Friedman Test

It is a statistical non-parametric test which is used to compare performances of different techniques applied on multiple datasets. The result of the test tells whether performances of techniques are statistically equivalent or not. It is not necessary that population is normally distributed. It is based on chi distribution where n-1 is degree of freedom when there are n techniques and hypothesis are checked at p-value=0.05

Followings are the hypothesis of the test:

H_0(Null): The performances of various software vulnerability prediction techniques are statistically same.
H_a: The performances of various software vulnerability prediction techniques are statistically different.

The performances of different techniques are compared with f-measure and Friedman test is applied in R code.

Friedman rank sum test

```
data:  Sample
Friedman chi-squared = 17.778, df = 6, p-value = 0.00681
As p-value <0.05
```

So, Null hypothesis is rejected. Hence there is statistical difference between the performances of different techniques

Since performance of different techniques are statistically different wilcoxon test is used to compare pair wise performance of different techniques

```
Wilcoxon signed rank test
data:  dataset$SVM and dataset$GW.RF
V = 0, p-value = 0.125
alternative hypothesis: true location shift is less than 0
Pair wise comparisons are performed between all techniques and
p-values are greater than 0.05
```

Hence there are no pair of techniques which are statistically different in performance.

8. APPLICATION OF THE WORK

Results of this study can be useful for both software security expert and software professionals as well as researchers in the following way:

- A tester can use resources effectively by identifying vulnerable classes and can detect security flaws in the system in the early stage of software development.
- Software developers can take decisions regarding the optimal use of resources available. They can decide which class requires larger resources to be non-vulnerable.
- Different techniques that are used for vulnerability prediction come from machine learning. This study suggests the efficiency of the technique in the detection of vulnerable software and proposed a novel technique which better than other machine learning techniques.
- Classes may be redesign in such a way that they reduce the vulnerability of classes. For example, if a software metric is found strongly related to the vulnerability then the designer can redesign the class to reduce that metric
- Metrics used in this study can be used in the software industry and by security experts to set quality benchmarks for different software organizations. An ideal range of metrics values can be set to check the software vulnerability and Metrics values of software products (similar to phpMyAdmin3.3, Moodle2, Drupal6) can be compared with the benchmarks set for software vulnerability if security experts or developers found that there is a large deviation from normal range then they can take the remedial step.

9. THREATS TO VALIDITY

In this section, we have discussed various threats to the validity of the study:

9.1 Construct Validity

This validity poses threats if the dependent and independent variables are not accurately measured (Zhou et al., 2009).There is a gap between the theoretical meaning of variables and actual measure. In our study, we used the data sets provided from the data repository however the severity of the vulnerability is not considered as it may be subjective.

9.2 Internal Validity

There is a threat of internal validity if there is a causal effect of independent variables on dependent variables. Controlled experiments are conducted to determine this effect. The possibility of the threat of internal validity in the study is present if software metrics are indirectly related to software vulnerability. The programmer's capability and experience are not considered while constructing the prediction model for software vulnerability prediction this may pose an internal validity problem.

9.3 External Validity

As external validity depends on up to which extent the results of a study can be generalized and in our study we have taken three data sets available in the literature. Data sets which are considered for vulnerability check are PHP based and there is a need to study other types of programming languages like (Java). So for generalization of results of our research, there is a need to explore different types of data from different application domains to replicate the study. Different parameters in the study are specified completely that increases the chance of generalizability of results.

10. CONCLUSION

Accurate prediction of the software vulnerability can mitigate the risk of software failure or unauthorized access to the system. Substantial efforts can be devoted to rectify the vulnerable software classes in the early stages of software development.

The main aim of this work is to investigate the performances of various MLT and metaheuristics based techniques for the detection of vulnerable software classes and Compared the performance of the MLT and metaheuristic based technique in the prediction of the vulnerable software.

The major findings of our work are as follows:

- Precision, recall, and f-measure of Metaheuristics based Random Forest are better compared to other MLT.
- Although Metaheuritic based Random Forest techniques are best as they uses optimization for features selection however these techniques consumes larger time compare to all other techniques
- Particle swarm optimization based random forest algorithm(PSO-RF) has given best results for software vulnerability prediction

Our future plan is to replicate our work on large data sets of different domains and we may improve the performance of used metaheurtistics techniques in the study for better results and novel metaheuristics technique can also be proposed for software security.

REFERENCES

Abunadi, I., & Alenezi, M. (2016). An empirical investigation of security vulnerabilities within web applications. *Journal of Universal Computer Science*, *22*(4), 537–551.

Alenezi, M. (2014). Fault-proneness of open source systems: An empirical analysis. *International Arab Conference on Information Technology*, 164–169.

Alenezi, M., & Abunadi, I. (2015). Evaluating Software Metrics as Predictors of Software Vulnerabilities. *International Journal of Security and Its Applications*, *9*(10), 231–240. doi:10.14257/ijsia.2015.9.10.21

Bansal, A. (2017). Empirical analysis of search based algorithms to identify change prone classes of open source software. *Computer Languages, Systems & Structures*, *47*, 211–231. doi:10.1016/j.cl.2016.10.001

Bardsiri, V. K. (2013). PSO-based a model to increase the accuracy of software development effort estimation. *Software Quality Journal*, *21*(3), 501–526. doi:10.1007/s11219-012-9183-x

Basili, V. R., Briand, L. C., & Melo, W. L. (1996). A validation of object-oriented design metrics as quality indicators. *IEEE Transactions on Software Engineering*, *22*(10), 751–761. doi:10.1109/32.544352

Bozorgi, M. (2010). Beyond Heuristics: Learning to Classify Vulnerabilities and Predict Exploits. *Proceeding KDD 16th ACM conference on Knowledge discovery and data mining*, 105-114. doi:10.1145/1835804.1835821

Burgess, C. J., & Lefley, M. (2001). Can genetic programming improve software effort estimation? A comparative evaluation. *Information and Software Technology*, *43*(14), 863–873. doi:10.1016/S0950-5849(01)00192-6

Carvalho, A. B. (2010). A symbolic fault-prediction model based on multi-objective particle swarm optimization. *Journal of Systems and Software*, *83*(5), 868–882. doi:10.1016/j.jss.2009.12.023

Catal, C. (2007). An artificial immune system approach for fault prediction in object-oriented software. *Proceedings of the second international conference on dependability computer systems*, 1–8. doi:10.1109/DEPCOS-RELCOMEX.2007.8

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. doi:10.1613/jair.953

Chowdhury, I., & Zulkernine, M. (2011). Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities. *Journal of Systems Architecture*, *57*(3), 294–313. doi:10.1016/j.sysarc.2010.06.003

Gyimothy, T., Ferenc, R., & Siket, I. (2005). Empirical validation of object– oriented metrics on open source software for fault prediction. *IEEE Transactions on Software Engineering*, *31*(10), 897–910. doi:10.1109/TSE.2005.112

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. *Proceedings of IEEE International Conference on Neural Networks*, 1942–1948. doi:10.1109/ICNN.1995.488968

Khalid, . (2018). *Predicting web vulnerability in web applications based on Machine learning, Intelligent Technologies and Applications. In INTAP 2018. Communications in Computer and Information Science*. Springer.

Malhotra, R. (2015). Empirical Research in Software Engineering: Concepts, Analysis, and Applications. CRC Press.

McGraw, G., & Potter, B. (2004). Software security testing. *IEEE Security and Privacy*, *2*(5), 81–85. doi:10.1109/MSP.2004.84

Meneely, A., & Williams, L. (2010). Strengthening the empirical analysis of the relationship between linus' law and software security. *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 1-10. doi:10.1145/1852786.1852798

Menzies, T., Greenwald, J., & Frank, A. (2007). Data mining static code attributes to learn defect predictors. *IEEE Transactions on Software Engineering*, *33*(1), 2–13. doi:10.1109/TSE.2007.256941

Minku, L. L., & Yao, X. (2013). Software effort estimation as a multi objective learning problem. *ACM Transactions on Software Engineering and Methodology*, *22*(4), 1–32. doi:10.1145/2522920.2522928

Mirjalili, S., Mohammad, S., & Lewis, M. A. (2014). Grey Wolf Optimizer. *Advances in Engineering Software*, *69*, 46–61.

Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT Press.

Neggaz, N., & Ewees, A.A., AbdElaziz, M., & Mafarja, M. (2020a). Boosting salp swarm algorithm by sine cosine algorithm and disrupt operator for feature selection. *Expert Systems with Applications*, *145*, 113103.

Neggaz, N., Houssein, E. H., & Hussain, K. (2020b). An efficient henry gas solubility optimization for feature selection. *Expert Systems with Applications*, *152*, 113364.

Pedregosa, . (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pendharka, P. C. (2010). Exhaustive and heuristic search approaches for learning a software defect prediction model. *Engineering Applications of Artificial Intelligence*, *23*, 34–40.

Rhmann, W., Pandey, B., & Ansari, A. A. (2021). Software effort estimation using ensemble of hybrid search-based algorithms based on metaheuristic algorithms. *Innovations Syst Softw Eng*. 10.1007/s11334-020-00377-0

Shar, L. K., Briand, L. C., & Tan, H. B. K. (2015). Web Application Vulnerability Prediction Using Hybrid Program Analysis and Machine Learning. *IEEE Transactions on Dependable and Secure Computing*, *12*(6), 688–707. doi:10.1109/TDSC.2014.2373377

Shin, Y., & Williams, L. (2008). Is complexity really the enemy of software security? *Proceedings of the Fourth ACM Workshop on Quality of Protection*, 47–50.

Shukla, K. K. (2000). Neuro-genetic prediction of software development effort. *Information and Software Technology*, *42*, 701–713.

Vandecruys, O. (2008). Mining software repositories for comprehensible software fault prediction models. *Journal of Systems and Software*, *81*, 823–839.

Vrbancic, G. (2018). NiaPy: Python microframework for building nature-inspired algorithms. *Journal of Open Source Software*, *3*(23), 1–2.

Walden, J. (2014). Predicting vulnerable components: Software metrics vs text mining. *IEEE 25th International Symposium on Software Reliability Engineering (ISSRE),* 23–33.

Zhang, Y., Lo, D., Xia, X., Xu, B., Sun, J., & Li, S. (2015). Combining software metrics and text features for vulnerable file prediction. *2015 20th International Conference on Engineering of Complex Computer Systems (ICECCS),* 40–49.

Zhou, Y. (2009). Examining the Potentially Confounding Effect of Class Size on the Associations between Object-Oriented Metrics and Change-Proneness. *IEEE Transactions on Software Engineering*, *35*, 607–623.

Wasiur Rhmann is an Assistant Professor at the Department of CSE at KL Deemed to be University, Green Fields, Vaddeswaram, A.P., India. Before Joining KL University he worked as an Assistant professor at Department of Computer Application at Shri Ramswaroop Memorial University (SRMU), Barabanki, Uttar Pradesh, India. Before joining SRMU he had worked as a Resource Person (Guest faculty) in the Department of Computer Science and Information Technology at Babasaheb Bhimrao Ambedkar University (A Central University), Satellite Campus Amethi, U.P., India. He did his Ph.D. (Computer Science) from Babasaheb Bhimrao Ambedkar University (A Central University), Lucknow, India. He is an alumnus of the prestigious Aligarh Muslim University, India, and completed his Master of Computer Application (M.C.A) and B. Sc. (Physics) from Aligarh Muslim University, Aligarh, India. He has published several research papers in international journals of repute. His research interests include Software testing, machine learning, software quality assurance, and UML modeling.